

Проект  
„Задача за търговският  
пътник“  
по  
Структури от данни

Изготвил:Йордан Павлов F№71978

## Съдържание:

### **Глава 1. Увод**

- 1.1. Описание на идеята и сложност на проекта
- 1.2. Структурата на документацията

### **Глава 2. Алгоритми – начин на работа, скорост, оценка на резултат**

- 2.1. Най-близък съсед
- 2.2. 2 opt

### **Глава 3. Графика на скорост на работа на алгоритмите и заеманата виртуална и физическа памет**

- 3.1. Сравнение на скоростта на най-близък съсед, 2-opt
- 3.2. Виртуална памет, физическата памет на алгоритъмът за най-близък съсед
- 3.3. Виртуална памет, физическата памет на алгоритъмът за 2-opt

### **Глава 4. Реализация, тестване**

- 4.1. Използвани структури от данни за реализацията на проблема
- 4.2. планиране, описание и създаване на тестови сценарии

### **Използвана литература**

## Глава 1.Увод

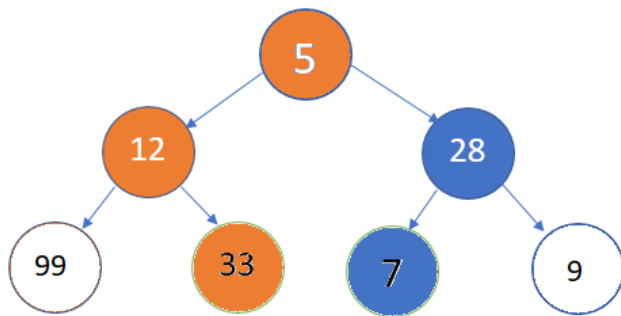
1.1 Търговският пътник има задачата да тръгне от определен град и да обиколи всички градове, който е предвидил за доставка на продукти и да се върне в града от където е тръгнал, като трябва да е мине по възможно най-краткият път. Проблемът е с NP сложност изисква проверка на резултатът. Евристиката представлява правене на извод, но няма гаранция ,че резултатът е оптимален.

1.2 Структурата на докоментацията се извършва по следните стъпки. Първата е да се представяне на належащите проблеми. Втората е търсенето на достатъчно ефективно решение. Документацията завършва с представянето на различни примери и сценарии с цел да се онагледят поведението на програмата в различните сценарии.

## Глава 2.Алгоритми

### 1. Най-близък съсед:

Този метод е един от първите алгоритми за решаване на проблема за търговския пътник. Търговският пътник тръгва от произволен град и многократно посещава най-близкия град, докато не обиколи всички градове. Алгоритъмът е сравнително бърз , обикновено не е оптимален. Сравнително дава средно път с 25% по-дълъг път от възможно най-краткия път, тоест това не е най-доброто решение. причината за този факт е алчността ,която се стреми да осигури бързи резултати ,а не преглежда по-големият набор от възможности. Един път взето решение, то никога не се преразглежда. Това ,че спестява стъпки пести и време на работа.



Начин на работа:

1. избиране на връх (зададен от потребителя).
2. винаги, когато достигне някакъв връх, поглежда тежестите на всички ръбовете, които водят до върхове, които все още не сте посетили и съответно избира този с най-малко тегло.
3. след като достигне последният връх се връща към началната си точка.

Скоростта на този алгоритъм е  $n^2$ .

## 2. 2-opt:

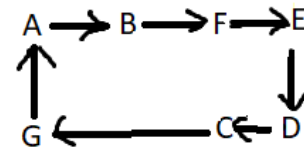
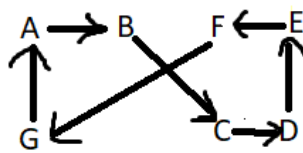
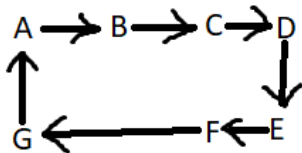
Този метод търсене на път със специално разменяне, което работи като негова евристика. Разменянето представлява избиране на 2 града примерно  $i$  и  $j$  премахваме двата пътя между  $i$  и следващият на  $i$  и  $j$  и следващият на  $j$ , като ги заменя с път между  $i$  и  $j$ , последващите на  $i$  и на  $j$ , като проверява дали новополученият път е по-кратък. ако пътят е по-дълъг продължаваме съответно без да премахваме пътища. Скоростта е  $n^2$ . На графиката показвам ,че времето за работа на този метод е експотенциален, по-бавен е от най-близкия съсед, но този алгоритъм показва максимално най-краткия път с малко отклонение.

Пример за път:  $A \Rightarrow B \Rightarrow C \Rightarrow D \Rightarrow E \Rightarrow F \Rightarrow G \Rightarrow A$

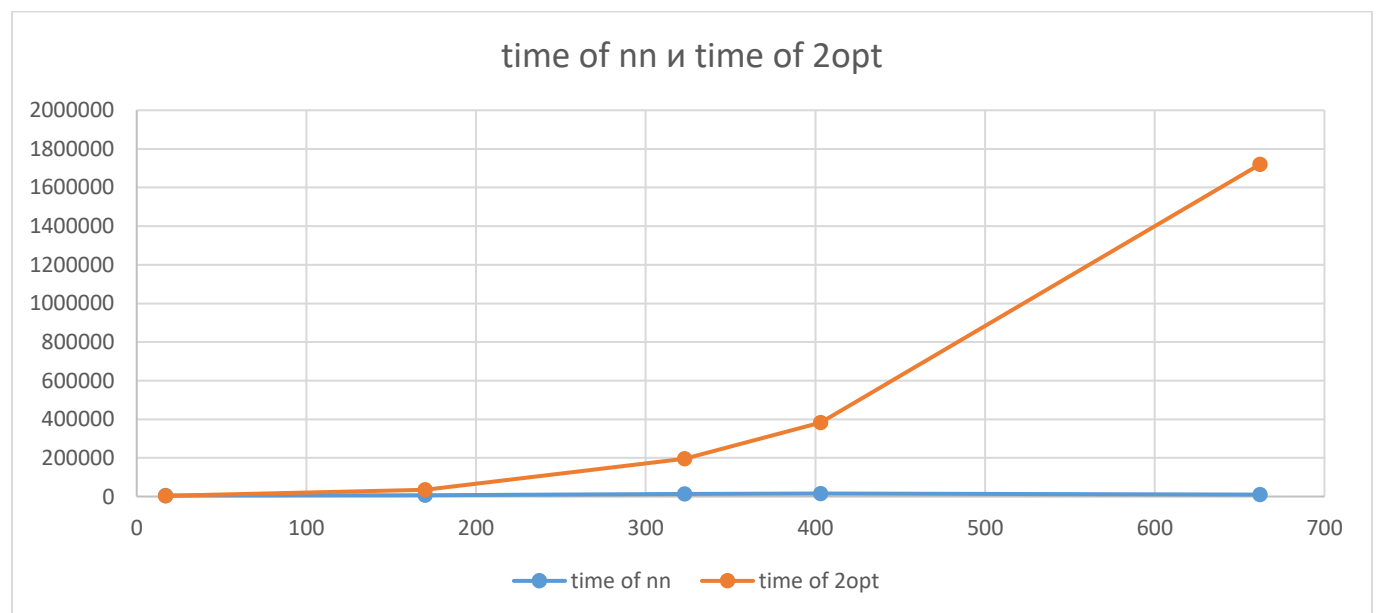
Избираме  $i = 3$  и  $j = 5$

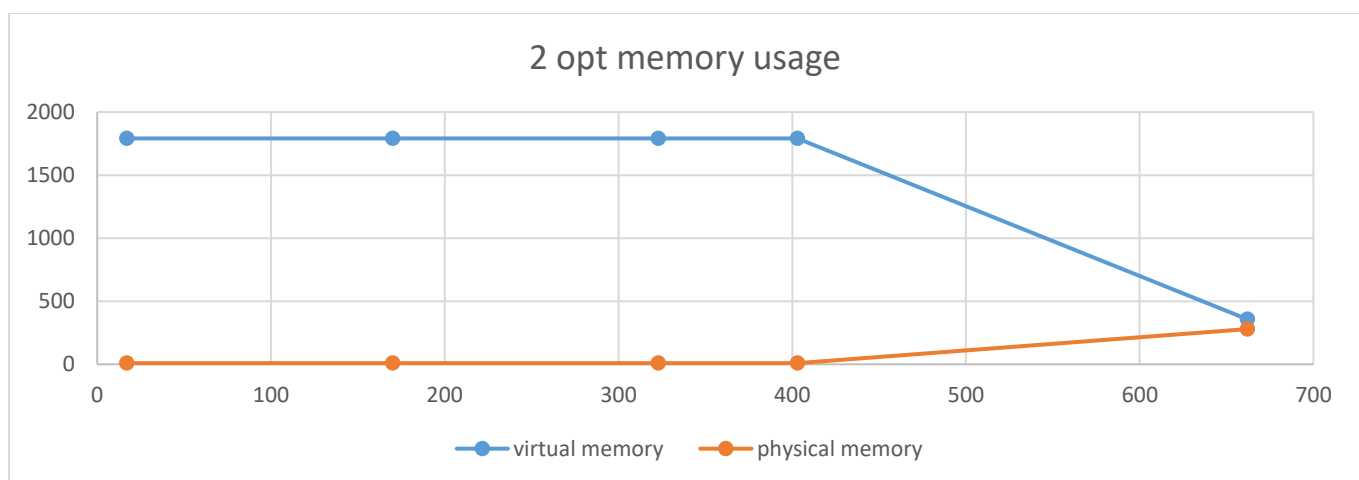
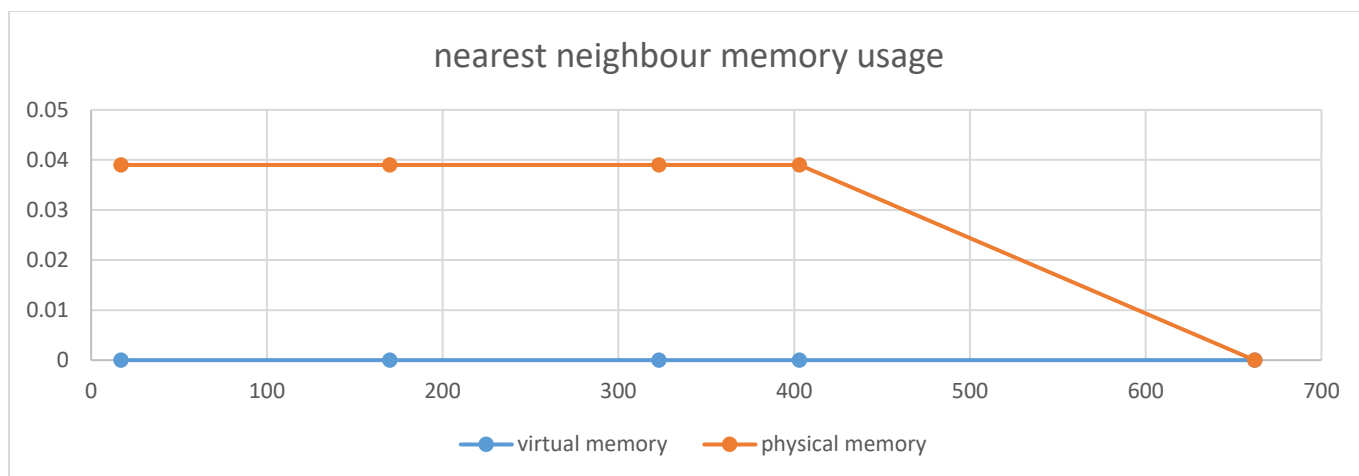
Новият път:

1.  $A \Rightarrow B$
2.  $A \Rightarrow B \Rightarrow F \Rightarrow E \Rightarrow D \Rightarrow C$
3.  $A \Rightarrow B \Rightarrow F \Rightarrow E \Rightarrow D \Rightarrow C \Rightarrow G \Rightarrow A$



## Глава 3.Графика на скорост на работа на алгоритмите и заеманата виртуална и физическа памет





**Пояснение:** Рязката разлика на графика 2 и 3 е от това ,че при последният тест прочитам кординати записвам ги като наредени двойки, съответно смятам разстоянието межу тях и го записвам в матрица. При останалите тестове получавам готова матрица която чета и записвам. Тестовите са 17 града(асиметрични), 170 града(асиметрични), 323 града(асиметрични), 403 града(асиметрични), 662 града(симетрични). Симетричните пътища пътищата са едноросочни, при асиметричните пътищата са двуросочни.

## Глава 4.Реализация, тесване

4.1. основната структура от данни която използвам е вектор причината е ,че има доста предимства, като `resize`, `reserve`, `clear`, `push_back`, `shrink_to_fit`. За четенето на точките използвам `pair` с което съхранявам x координати и y

координати. За най-близък използвам стек с цел да държа настоящият град, удобен е да тази цел.

4.2. Тестът с 17 града е избран с цел с по-малкото си градове да се провери дали вярно работят алгоритмите, тестът с 323 града е избран ,за да се покаже как се справя с повече градове, тестът 662 града е избран ,за да се покаже как се справя програмата с подадени координати.

когато се стартира програмата

name of file: br17.atsp

type of file matrix/points: matrix

from what point want to start: 10

изход на програмата: min weigth path is 43 path can be seen in perform\_opt.txt

name of file: ftv323.atsp

type of file matrix/points: matrix

from what point want to start: 30

изход на програмата: min weigth path is 3187 path can be seen in perform\_opt.txt

name of file: xql662.tsp

type of file matrix/points: points

from what point want to start: 0

изход на програмата: min weigth path is 3060 path can be seen in perform\_opt.txt

Използвана литеретура:

Heuristics for the Traveling Salesman Problem by Christian Nilsson.

<https://viktorgrigorov.wordpress.com/2017/02/26/290/>

<https://stackoverflow.com/questions/63166/how-to-determine-cpu-and-memory-consumption-from-inside-a-process>

<https://www.pluralsight.com/blog/software-development/how-to-measure-execution-time-intervals-in-c->